to be submitted manuscript No.
(will be inserted by the editor)

A Matrix-Based Evolutionary Algorithm for Cardinality-Constrained Portfolio Optimisation

Matthew J. Craven · **David I. Graham**

Received: date / Accepted: date

Abstract This article presents a matrix-based evolutionary algorithm to approximate solutions of the simultaneous multiple portfolio optimisation problem under cardinality constraints, for a selection of indices containing from $n = 31$ to $n = 493$ assets. This problem is made NP-hard by the requirement to find the best sub-portfolios of $k < n$ assets (in practice, $k \ll n$) from the vast number of possibilities and, simultaneously, the efficient frontier (EF) for these sub-portfolios. We study algorithm performance under a spread of cardinality constraint values, finding that there exists a small subset of $k < n$ assets for a given dataset with which it is possible to obtain a close approximation of the unconstrained EF. Computation times can be significantly reduced using this trick. Finally, by pooling results from a number of independent realisations and employing a sifting algorithm to the pooled results, we obtain significantly improved estimates of the EFs for the cardinality-constrained problem.

Keywords Stochastic · Evolutionary algorithm · Finance · Portfolio optimisation · Cardinality constraint

Mathematics Subject Classification (2010) 68T20, 68W20, 90B50

1 Introduction

The Portfolio Optimisation Problem (POP) has been an important problem in financial mathematics since the work of [19]. An investment portfolio is a collection of financial assets held by an investor over a period of time; the investor may be an individual, finance house, hedge fund or other entity. The standard paradigm in Modern

M. J. Craven
Centre for Mathematical Sciences, Plymouth University, PL4 8AA, UK.
E-mail: matthew.craven@plymouth.ac.uk ORCID: 0000-0001-9522-6173

D. I. Graham
Centre for Mathematical Sciences, Plymouth University, PL4 8AA, UK.
E-mail: d.graham@plymouth.ac.uk ORCID: 0000-0002-6084-5676

Portfolio Theory states that the investor wishes to minimise risk while maximising return over the lifetime of their investment. By its nature, this is a tradeoff of risk against return. Depending upon the model, the classical model being a mean-variance model, this tradeoff may be controlled by a so-called risk aversion parameter which represents the risk attitude of an investor. The collection of maximum returns for given levels of risk (or, alternatively, minimal risks for given levels of return) is referred to as the efficient (or Pareto) frontier.

1.1 Literature Review

There has been a large amount of work on the POP and its variants. The work of [19] formulated the POP as a quadratic optimisation problem, meaning that the problem may be solved by quadratic programming (QP) [15]. However, this formulation is rather basic, and further constraints are often added to make the model more realistic. In an effort to remedy some of these criticisms, more realistic variations of the POP have been proposed. For example, cardinality constraints or minimum lots constraints may be added. However, these inclusions often render the modified POP NP-complete [18] and hence make the problem more interesting from a computational point of view.

A large part of the complexity of these modified POPs is the vast number of combinations for selectable assets, based upon the restrictions which are applied. In these situations a very large search space is typical, meaning that deterministic approaches will be unsuccessful. Work often then focusses on stochastic or heuristic algorithms which introduce randomness into an approach to address "sticking points".

The Cardinality-Constrained POP (CCPOP), where a subset of k assets is selected from a total of n assets, has been well-studied using a variety of heuristics. The work of [7] surveyed three heuristic algorithms (genetic algorithm, simulated annealing and tabu search), adding a minimum proportion constraint and illustrating performance differences in the algorithms when finding the efficient frontier. Also, [12] compared three evolutionary algorithms with a Sharpe's Index objective and examined dynamics of the R^2 and hypervolume indicators; [26] briefly investigated convergence of a hybrid GA-local search algorithm and an Ant Colony Optimiser on the CCPOP with shorting, large k, limited numbers of assets, bounded tracking error volatility, with the objective of maximising Sortino Ratio. The work of [25] followed with a study of evolutionary algorithm convergence properties on the CCPOP with floor/ceiling constraints, maximising the Sortino Ratio. Additionally, [6] attempted to solve the CCPOP with small minimum proportions via an "Increasing Set Algorithm", exhibiting broadly increasing times to produce solutions of increasing cardinality.

Related is the CCPOP with minimum lots. This is where assets are purchased in multiples of minimum standard amounts, lots (also known as buy-in thresholds), of each asset. The work of [18] investigated this problem using three mixed integer linear programming algorithms. Subsequently, [17] used an approach combining differential evolution with machine learning (MODEwAwL) to a cardinality-, floor- and ceiling-constrained POP with minimum lots, testing the algorithm against the multi-objective evolutionary algorithms NSGA-II, SPEA2, PESA2 and the '1+1' local search evolutionary strategy PAES. Their approach used the datasets of OR-

Library [2] and relatively large cardinality constraints. Further, [1] conducted a comparison of five MOEAs on the CCPOP with floor constraints, using, among other datasets, those of [2] as above. We consider this problem out of scope, but present the above references due to their importance in the field.

Heuristic approaches, such as many of the above, generally use a vector representation of solutions. However, there was a short work by [10] that experimented with a matrix representation for solving the cardinality-constrained POP over many different time intervals, showing how proportions of assets in portfolios may change over time. The work compared an evolutionary algorithm with a hillclimber algorithm, showing that in cases of local optima of the objective function the former algorithm with this representation had a clear advantage over the hillclimber. Other approaches to solving POP variations may be had from stochastic analysis [11], physics [23], time-series prediction [20], dynamic stochastic programming [22] and parametric quadratic programming [14].

1.2 Outline of the Work

In this paper, an innovative evolutionary algorithm-regression scheme algorithm for approximating solutions to the cardinality-constrained multiple POP (CCMPOP) is firstly presented. A matrix representation is used in the approach, allowing simultaneous solution (rather than individual or sequential solution) of a sequence of POPs and efficient discrimination between candidate solutions for individual points on the efficient frontier (Section 3). Extensive testing of the algorithm on real-world and benchmark datasets up to 493 assets is then performed, showing performance consistent with other treatments of the problem and demonstrating its validity.

Subsequently, the number ($k_{\min} < n$) of assets in each dataset that are the "most important" is identified by solving the unconstrained POP using QP. We find that the EA typically requires a greater number of assets (we call this $u_{\min}$ in the following) to achieve a good approximation of the unconstrained QPEF. However, the computational cost of finding a good approximation to the unconstrained EF using a cardinality constrained EA can be significantly less than using an EA without the cardinality constraint. Finally, by pooling results from a number of independent realisations and employing a sifting algorithm to identify 'non-dominated' points, we obtain significantly improved estimates to the cardinality-constrained efficient frontier (i.e., the optimal solution) and illustratively compare our results with the existing solutions of [4]. The work is illustrated by over 1500 test runs, copious analyses of the convergence and results of the runs, the metrics used and subsequent conclusions. Our treatment begins in the next section, defining the key mathematical terms and the problem.

2 Mathematical Preliminaries

In the following two subsections, only pertinent concepts are detailed. Interested readers are referred to the work of [1, 7, 17], for example, for further details.

2.1 Notation

Suppose we have n assets. A portfolio is represented by a vector of weights $\mathbf{w} = (w_1, \ldots, w_n)$, with the condition that $\sum_{i=1}^{n} w_i = 1$. Each weight w_i is the proportion of the total money of the investor that is invested in asset i, and we assume that each weight is non-negative. Let $\mathbf{X}$ denote the matrix of historical returns from each asset for a given time period of p intervals (e.g., daily, monthly). This takes the form of a $p \times n$ matrix where each row corresponds to a distinct sub-period and each column to a distinct asset. Let $\mu = \left(\frac{1}{p} \sum_{i=1}^{p} X_{ij} \right)_{j=1}^{n}$ denote the vector of mean returns, over the given time period, for all assets $j = 1, \ldots, n$ and σ denote the variance-covariance matrix of $\mathbf{X}$. The expected return (scalar) from $\mathbf{w}$ is the scalar product

$$R = (\mathbf{w}, \mu). \tag{1}$$

The risk (scalar) of a portfolio vector $\mathbf{w}$ is the variance

$$V = \sum_{i=1}^{n} \sum_{j=1}^{n} w_i w_j \sigma_{ij} \tag{2}$$

The Markowitz classical formulation of the POP is [19]:

Minimise V subject to

$$\begin{aligned} R &\geq R_0 \\ \sum_{i=1}^{n} w_i &= 1 \\ w_i &\geq 0 \text{ for } i = 1, \ldots, n \end{aligned} \tag{3}$$

This is also known as the unconstrained problem. This problem is solved via QP, minimising risk V for a portfolio subject to a given return $R \geq R_0$. The solution is of polynomial-bounded complexity in the number of assets, n. It is easy to show that, by removing the constraint that $w_i \geq 0$, the problem admits a closed-form analytic solution [3]. However, this is not the case for $w_i \geq 0$. In general, solving via QP is possible as long as the objective function is quadratic and the variance-covariance matrix is invertible.

Often, the Markowitz classical formulation is converted to a risk-parameterised version as follows:

Find a weight vector $\mathbf{w}$ with $\lambda V - (1 - \lambda) R$ minimised subject to

$$\begin{aligned} \sum_{i=1}^{n} w_i &= 1 \\ w_i &\geq 0 \text{ for } i = 1, \ldots, n \end{aligned} \tag{4}$$

Proof of this result may be found in [7]. Note that the $R = R_0$ constraint no longer appears. In this formulation, return values are found as part of the computation. Note

also that λ values must lie between 0 (to maximise the return) and 1 (to minimise the risk). The POP is a continuous optimisation problem due to the possible values of the assigned weights. Adding the cardinality constraint

$$\#\{i : \mathbf{w}_i \neq 0\} \leq k,$$

to restrict the number of non-zero asset weights used to a maximum of k, transforms problem (4) into the cardinality-constrained POP. In the next subsection we detail the cardinality-constrained multiple POP.

2.2 Expansion into a Sequence of POPs

Taking the problem of the previous subsection, we wish to solve the problem for m values of λ simultaneously. This means that our algorithm weight representation needs to be generalised from a weights vector, $\mathbf{w}$. To do this, we introduce a *weights matrix*. A weights matrix $\mathbf{W}$ is an $m \times n$ matrix with each of the m rows being a weight vector, written as follows:

$$\mathbf{W} = \begin{pmatrix} w_{11} & w_{12} & \cdots & w_{1n} \\ w_{21} & w_{22} & \cdots & w_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ w_{m1} & w_{m2} & \cdots & w_{mn} \end{pmatrix}$$

Each row of $\mathbf{W}$, denoted by $\mathbf{W}(i,:)$, is indexed by a distinct value of i (and so λ).

In turn, the measures of risk and return used must change with this change in representation. Hence the expected return vector from the weights matrix $\mathbf{W}$ is

$$\mathbf{R} = \mathbf{W}\mu^T. \tag{5}$$

The risk is also a vector, with each entry V_i for $i = 1, \ldots, m$ given by

$$V_i = \mathbf{W}(i,:)\sigma\mathbf{W}(i,:)^T. \tag{6}$$

We wish to effectively approximate solutions of the cardinality-constrained POP for m values of λ taking the range $[\lambda_{\min}, \lambda_{\max}]$ (details on the determination of this range are given in Section 3.4). The objective function is

$$C(\mathbf{W}) = \frac{1}{m}\sum_{i=1}^{m}\left(\lambda_i V_i - (1-\lambda_i)R_i\right), \tag{7}$$

where i represents the index of a given portfolio on the solution curve (see below). The λ-formulation of the multiple cardinality-constrained POP (recall, denoted the CCMPOP) is then written similarly to (4) as:

Find a weight matrix $\mathbf{W}$ with *minimal* objective value $C(\mathbf{W})$ subject to

$$\begin{aligned}
&\sum_{j=1}^{n} \mathbf{W}(i,:) = 1 \text{ for } i = 1,\ldots,m \\
&\mathbf{W}(i,j) \in [0,1] \text{ for } i = 1,\ldots,m, \quad j = 1,\ldots,n \\
&\#\{j : \mathbf{W}(i,j) \neq 0\} \leq k \text{ for } i = 1,\ldots,m
\end{aligned} \tag{8}$$

The final condition, of course, means that there are at most k non-zero weights in any selected portfolio (i.e., row of $\mathbf{W}$). A risk-return pair (V_i, R_i) for a given portfolio is *efficient* if, for the risk V_i, there does not exist any other portfolio which has a larger return. The collection of risk-return pairs (V_i, R_i) which are efficient for the values λ_i ($i = 1,\ldots,m$) is referred to as the *efficient frontier*. In particular, for the unconstrained problem (where $k = n$) we have the unconstrained efficient frontier (UEF or QPEF). When $k < n$ the efficient frontier is termed the cardinality-constrained efficient frontier (CCEF). The number m refers to the number of points on the CCEF, i.e. the number of different λ values used. Note we consider the CCMPOP as a single-objective problem (in λ); algorithms to treat the problem as a multi-objective problem exist but are out of scope of this work. The next section describes the algorithm used to approximate solutions to the CCMPOP.

3 Algorithm

3.1 Evolutionary Algorithms

Evolutionary algorithms (EAs) are population-based stochastic optimisers which use neo-Darwinian laws of nature to manufacture solutions to a problem in an iterative way. Such algorithms are useful for problems which have especially large (possibly infinite) search spaces, and have been successfully used in hard problems (which may mean NP-complete, but we do not restrict to this complexity class). In the real world, this type of algorithm is used in Civil Engineering [13], improved designs for circuits [16], and cryptography [9], to name but a few disciplines.

EAs begin with an initial generation of candidate solutions to a chosen problem, typically chosen at random (although many methods exist). These candidate solutions are bred to create subsequent generations of other candidate solutions. The processes of evolution are mimicked by evolutionary operators, of which three kinds are typically used: *selection* ensures that high-performing candidates are carried through to the next generation; *mutation* performs small changes to candidates in order to fine-tune them against some performance measure; finally, *crossover* performs large changes to a candidate and is most useful when the progress of evolution simulated by the algorithm is trapped in a local optimum. Some other operators may also be used, such as random immigration, in order to balance population diversity and control the likelihood a local optimum will occur.

There are advantages and disadvantages associated to EAs. The main advantage is that the population-based structure of an EA enables efficient search of large sections of search spaces. Multiple solutions may also be found. In addition, EAs are classically black-box methods which do not depend very much upon the structure of

the problem or search space (although, recently, research has been moving in the opposite direction, making such methods easily applicable on, and easily transferrable between, problems). A disadvantage is that EAs are acknowledged to be sensitive to algorithm parameter settings and, hence, their performance is variable. In addition, local extrema of the objective function often, in practice, cause problems for optimisation. However, this is typically dependent upon search space structure and various tools for mitigation of this issue do exist: such a tool is the 'guidance' algorithm developed in Section 3.3. Next, we present the operators used in our work.

3.2 Algorithm Operators

The algorithm is matrix-based, with each population individual a weights matrix, meaning that the operations are also matrix-based. The following three types of operators act upon weights matrices in the "current generation" to produce weights matrices in the subsequent generation. Recall that m is the number of points on the CCEF approximation (and so the number of rows of the weights matrix $\mathbf{W}$).

Selection is partially elitist and by 3-tournament. Copy the weight matrix $\mathbf{W}^{(1)}$ with minimal objective value from the population, and then matrices $\mathbf{W}^{(i_1)}, \ldots, \mathbf{W}^{(i_s-1)}$ chosen by 3-tournament, to the next generation. In this case, a tournament means that three matrices are chosen uniformly at random (u.a.r.) from the population, and the matrix with the lowest objective value (as we are attempting to minimise the objective function) is selected.

Mutation is also partially elitist. For the first execution, choose the matrix $\mathbf{W} = \mathbf{W}^{(1)}$ with minimal objective value from the population. Take the focus row, q, and its nearest neighbours q_1, q_2 (identified by Algorithm 4). Then let

$$\mathbf{W}(q,:) \leftarrow \alpha_m \mathbf{W}(q,:) + \frac{1-\alpha_m}{2} \left(\mathbf{W}(q_1,:) + \mathbf{W}(q_2,:) \right). \tag{9}$$

For subsequent executions, choose $\mathbf{W}$ from the top 10% of the population by least objective value first order. Take $q \in \{1, \ldots, m\}$ uniformly at random (u.a.r.) and then set $\mathbf{W}(q,:) = \tau \mathbf{W}(q,:)$, where τ is the stochastic vector perturbation given by Algorithm 3. Either operation performs mutation to a single row of $\mathbf{W}$, and the resulting matrix is put into the next generation.

Crossover is a parameterised diploid crossover producing two offspring. Choose two matrices $\mathbf{W}_1$, $\mathbf{W}_2$ u.a.r. from the top 20% of population by rank. Then compute $\mathbf{W}_1^c = \alpha_c \mathbf{W}_1 + (1-\alpha_c)\mathbf{W}_2$ and $\mathbf{W}_2^c = \alpha_c \mathbf{W}_2 + (1-\alpha_c)\mathbf{W}_1$. For each execution of crossover, matrices $\mathbf{W}_1^c, \mathbf{W}_2^c$ are put into the next generation.

Control parameter values were chosen by experiment to provide acceptable performance and computation time. We chose the mutation parameter $\alpha_m = 1/2$ and the crossover parameter $\alpha_c = 1/3$. For convenience, the number of population elements produced by the algorithm was set to two elements by selection, 20 by crossover, and 28 by mutation, giving an overall population size of fifty. This population size was used for speed (and is not influenced by the value of m). Each generation of the EA computes an approximation of the EF (over m points) that is closer to the true EF than that of the previous generation.

3.3 The Main Algorithm

The EA (Algorithm 1) aims to find approximate solutions to problem (8) and is guided by the objective function (7). The code in the present work, written in MATLAB, substantially builds upon that from previous work of the first author [10]. In both the present and previous work, the EA population consists of weights matrices, each a candidate solution to the CCMPOP, and the EA output is a weights matrix with the best objective value found after a given number, g_0, of generations. The current work builds upon the previous methodology but incorporates distinct operators and improves upon it by the adding repair, perturbation and guidance algorithms described in this section, as well as performing much more comprehensive analyses of the results produced.

Algorithm 1 Main Algorithm

Input: population size p, returns matrix $\mathbf{X}$, initial weight matrix $\mathbf{W}_0$, number of populations to run EA g_0, $\lambda_{\min}$
Output: a weight matrix $\mathbf{W}^*$ of best objective value found

$g \leftarrow 1$
generate initial population as follows. Take the first member of the population as $\mathbf{W}^{(1)} = \mathbf{W}_0$, the given initial weights matrix
for all $j = 2, \ldots, p$ **do**
    $\mathbf{W}^{(j)} = [\pi(\tau^s \mathbf{W}(1,:)),,\pi(\tau^s \mathbf{W}(m,:))]$ for a random permutation π, and the perturbation operation τ according to Algorithm 3
end for
while $g < g_0$ **do**
    check feasibility of all population members $\mathbf{W}^{(1)}, \ldots, \mathbf{W}^{(p)}$ according to (8)
    repair infeasible population members by Algorithm 2
    compute objective values $C(\mathbf{W}^{(1)}),,C(\mathbf{W}^{(p)})$ for all $\lambda \in [\lambda_{\min}, 1]$
    rank in minimal objective value first order:

$$C\left(\mathbf{W}^{(i_1)}\right) \leq C\left(\mathbf{W}^{(i_2)}\right) \leq \ldots \leq C\left(\mathbf{W}^{(i_p)}\right)$$

    execute operators according to the EA control parameter values to population $g+1$
    $g \leftarrow g+1$
end while
return the least objective value weight matrix, $\mathbf{W}^* = \mathbf{W}^{(1)}$

Algorithm 2 is a probabilistic repair algorithm for infeasible candidate solutions, which, in practice, performs well. If a candidate is not repaired within a given number of times repair is attempted then it is accepted for inclusion into the generation and, due to its lack of feasibility is likely then not selected for the next generation (as it will not be well-performing).

Algorithm 2 Repair

input: Matrix $\mathbf{W}$, limit
output: d_f, repaired matrix $\mathbf{W}$
1: record the feasibility degree, d_f, as the sum of the number of times each row of the matrix $\mathbf{W}$ violates the conditions of problem (8)
2: for $g > 1$ record the violation degree vector, $(d_v)_{i=1}^m$, with elements $d_{v_i} = \begin{cases} 1 & \text{if } \#\{\mathbf{W}(i,:) \neq 0\} > k \\ 0 & \text{otherwise} \end{cases}$
3: **for all** i such that $d_{v_i} = 1$ **do**
4: $s \leftarrow \#\{\mathbf{W}(i,:) \neq 0\} - k$
5: $\mathbf{v} \leftarrow \{j : \mathbf{W}(i,j) \neq 0\}$ ▷ positions in row i of $\mathbf{W}$ of non-zero values
6: randomly permute $\mathbf{v}$ (i.e., $\mathbf{v} \leftarrow \pi\mathbf{v}$)
7: $c \leftarrow 0$
8: **for all** $j = 1, \ldots, s$ **do**
9: $c \leftarrow c + \mathbf{W}(i, \mathbf{v}[j])$
10: **end for**
11: let j_1 be a random position from the set $\{1, \ldots, n\} \setminus \mathbf{v}$
12: $\mathbf{W}(i, j_1) \leftarrow \mathbf{W}(i, j_1) + c$
13: **end for** ▷ thus there is a maximum of k assets from n chosen.
14: continue with step (2)

Algorithm 3 applies a random perturbation s times to rows identified in Algorithm 2. By experiment, we found the perturbation strength $s = 50$ was appropriate.

Algorithm 3 Stochastic Vector Perturbation (Restriction of [8])

input: weight vector $\mathbf{v}$ of length N, perturbation strength s
output: s-perturbed weight vector $\tau^s\mathbf{v}$ of length N
1: $s_0 \leftarrow 0$
2: **while** $s_0 \leq s$ **do**
3: choose $j_1, j_2 \in \{1, \ldots, N\}$ and $\delta \in [0, \min(\mathbf{v}[j_1], \mathbf{v}[j_2])]$ u.a.r.
4: $\mathbf{v}[j_1] \leftarrow \mathbf{v}[j_1] - \delta, \quad \mathbf{v}[j_2] \leftarrow \mathbf{v}[j_2] + \delta$
5: $s_0 \leftarrow s_0 + 1$
6: **end while**

To accelerate convergence of risk-return pairs to an efficient frontier, and assist in cases of local minima objective values, the Guidance Algorithm (algorithm 4) is applied once in each round of mutations per population. The algorithm guides the EA to solutions of the correct form by moving points closer to a fitted quadratic. Firstly, a quadratic, F, is fitted to the collection of portfolios associated to the best-performing weights matrix, $\mathbf{W}^{(1)}$, in the current population. An example is presented in Figure 1.

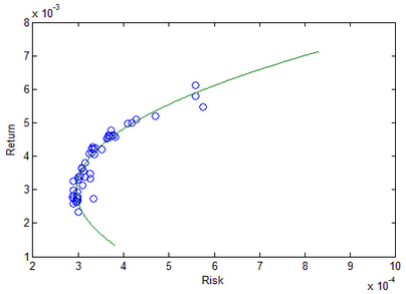


Fig. 1: An example of an interpolated quadratic, F, produced using the MATLAB `polyfit` command, through the risk-return pairs (circles), for fifty values of λ.

In step 7 of the Guidance Algorithm the risk-return pair (point) $\mathbf{z} = (V_j, R_j)$ has a probability of being chosen by roulette wheel proportional to its horizontal distance from F. The point, $\mathbf{z}$, chosen identifies with some row, $q \leq m$, of the weights matrix $\mathbf{W}^{(1)}$. After this, transformation (9) is applied to row q of that weights matrix.

Algorithm 4 Guidance Algorithm

input: $T = \{(V_i, R_i) : i = 1, \ldots, n\}$, the set of risk-return pairs
output: a point $\mathbf{z} = (V_j, R_j)$ of distance d_j from the curve F, and its nearest neighbours $\mathbf{m}_1$, $\mathbf{m}_2$
via quadratic regression, fit a quadratic curve F to points in T
for all i=1,...,n **do**
  $d_i \leftarrow V_i - F(V_i)$ ▷ horizontal distance from each point in T to F
end for
$I \leftarrow \{i : d_i > 0\}$ ▷ ordered set, with identical ordering to those of the $\lambda \in [\lambda_{\min}, 1]$
$D \leftarrow \{d_i : i \in I\}$ ▷ also an ordered set
select a point, $\mathbf{z} = (V_j, R_j)$ $(j \in I)$ by roulette wheel selection
$q \leftarrow j$ ▷ q is the index of point $\mathbf{z}$ in the sequence of points T and d_q is the horizontal distance from F
$E \leftarrow \{1, \ldots, m\} \setminus I$ ▷ set E is the set of points in $\{1, \ldots, m\}$ with the set I removed
if $q = 1$ **then** ▷ choose nearest neighbours
  $\mathbf{m}_1, \mathbf{m}_2 \leftarrow (V_2, R_2)$
else if $1 < q < m$ **then**
  $\mathbf{m}_1 \leftarrow (V_{i_1}, R_{i_1}) \in E$ with $i_1 = \max(i : i < j)$
  $\mathbf{m}_2 \leftarrow (V_{i_2}, R_{i_2}) \in E$ with $i_2 = \min(i : i > j)$
else
  $\mathbf{m}_1, \mathbf{m}_2 \leftarrow (V_{m-1}, R_{m-1})$
end if

To show the efficacy of the Guidance Algorithm (algorithm 4), the EA was run on dataset (D3) ($n = 89$, see next subsection) with cardinality $k = 2$. The EA was run ten times without guidance, and ten times with guidance. The root mean-square error

$$RMSE = \sqrt{\left(\frac{1}{m} \sum_{i=1}^{m} |(V_{QP,i}, R_{QP,i}) - (V_{EA,i}, R_{EA,i})|^2 \right)}, \tag{10}$$

between the efficient frontier found by the MATLAB `quadprog` QP solver and that obtained by the EA after $g = 2.5nm = 11125$ generations. This is the root mean-square of all distances between (r_i, R_i) pairs on the EA solution associated to given λ (indexed by i) and corresponding QP points. Obtained results are shown in Table 1.

Table 1: Comparative results for EA tests on instance (D3) with $k = 2$ without and with Algorithm 4. The final row shows the results of a comparison using the measure in the relevant column.

Alg. 4?	Mean objective value reached	99.9% conv. gen.	Mean exec. time (s)	RMSE
Without	-0.0014139	2475.7	4156.1	0.0010698
With	-0.0015359	1683.5	1602.9	0.00087021
Comparison	+	+	+	+

Observe that the EA with guidance is more effective on every measure than without. The mean objective value (which the EA seeks to minimise) is smaller, the number of generations until 99.9% convergence is smaller (although, this, of course may be convergence to a different, possibly local, minimum), and the mean execution time and RMSE are smaller. For more on the above measures, see Section 4.1.

3.4 Datasets Used

We used the following datasets to test EA accuracy and efficiency. Each dataset is in the form of historical returns on assets generated from asset prices at regular intervals between the dates shown.

1. S&P389: derived from daily prices of S&P500 constituent assets from 18/11/1999 to 09/08/2013 [21]. Only assets that were constituents for the complete period were included, giving 389 assets taken over 3926 timesteps.
2. S&P493: derived similarly to the above. This time, only assets that were constituents during the 1-year period 10/08/2012 to 09/08/2013 (250 timesteps) were included, giving an increased number of 493 assets.
3. The datasets (D1)-(D5) of OR-Library [2], which feature 31 (Hang Seng), 85 (DAX 100), 89 (FTSE 100), 98 (S&P 100) and 225 (Nikkei) assets over 290 weekly time intervals respectively.

The covariance matrices and returns for the first two (S&P) datasets are available at `http://math-sciences.org/datasets`.

3.5 Ranges of λ

As noted above, the EF is parameterised by λ. Thus λ values must lie between 0 (to maximise return) and 1 (to minimise risk). In some cases, however, a range of values of λ will lead to the same (risk, return) point on the efficient frontier. Furthermore, it is not possible to predict *a priori* the (V,R) pair on the efficient frontier associated with a given value of λ. We thus determined a 'useful' range of values of λ over which there was no duplication of coordinates. This range depends upon dataset used. Clearly, the maximum value of λ is always 1 as this corresponds to risk minimisation. The minimum value of λ can be determined *a priori* as follows:

The gradient $\frac{dR}{dV}$ at any point on the EF is given by $\frac{\lambda}{(1-\lambda)}$ (see [7]), which decreases with decreasing λ. Thus the minimum value of λ can be determined by finding the minimum gradient, which is found at the point on the EF for which return is maximised. The gradient at this point is determined by considering a 2-asset subportfolio consisting of the two assets with the highest returns. Suppose these assets have returns R_1 and R_2 and weights w_1 and $1-w_1$ respectively in the 2-asset portfolio. Assuming the covariances between them it may be shown that the risk is

$$V = \begin{pmatrix} w_1 & 1-w_1 \end{pmatrix} \sigma \begin{pmatrix} w_1 & 1-w_1 \end{pmatrix}^T = \sigma_{11} w_1^2 + 2\sigma_{12} w_1 (1-w_1) + \sigma_{22} (1-w_1)^2 \tag{11}$$

with return

$$R = w_1 R_1 + (1-w_1) R_2. \tag{12}$$

The gradient of the EF is given by,

$$\frac{dR}{dV} = \frac{dR}{dw_1} \Big/ \frac{dV}{dw_1} = \frac{R_1 - R_2}{2\sigma_{11}w_1 + 2\sigma_{12}(1-2w_1) + 2\sigma_{22}(1-w_1)}. \tag{13}$$

The gradient is minimised when $w_1 = 1$. Evaluating the above derivative at this point,

$$\left.\frac{dR}{dV}\right|_{w_1=1} = \frac{R_1 - R_2}{2(\sigma_{11} - \sigma_{12})}. \tag{14}$$

As noted above, the gradient of the EF is $\frac{\lambda}{1-\lambda}$. Equating this with (14) gives

$$\lambda_{\min} = \frac{1}{1 + 2\frac{\sigma_{11}-\sigma_{12}}{R_1-R_2}}. \tag{15}$$

The useful λ ranges $[\lambda_{\min}, 1]$ calculated in this way are displayed in Table 2. Note that the calculated values were also confirmed numerically. All EA experiments, except where noted, were performed with respect to the range of λ values specified for each dataset.

Table 2: Ranges of λ values for each dataset.

Dataset	D1	D2	D3	D4	D5	S&P389	S&P493
λ range	[0.342,1]	[0.149,1]	[0.426,1]	[0.107,1]	[0.115,1]	[0,1]	[0.489,1]

4 Testing Environment and Regime

In this section we detail the testing environment and regime under which we ran EA experiments. All tests in this section were performed on an i7 4-core laptop running at 2.2GHz and using 8 GB of memory. To illustrate the correctness of solutions obtained, these solutions were compared with the standard quadratic programming solutions using several metrics. The metrics used to perform this comparison and judge the success of the EA were as follows.

4.1 Testing Setup

Measuring the success of an algorithm on a continuous optimisation problem is a non-trivial task, each metric having advantages and disadvantages. Indeed, it is generally agreed that there may be no perfect measure of EA convergence in this case. Hence we detail below several metrics of algorithm success and convergence that we shall use, each produced by examination of multiple EA runs across all datasets (see Figure 5b for a typical example of convergence).

1. Number of generations, g_p, until 99.9% of the final objective value, C_g, is reached for a population of size p.[1]

[1] It is acknowledged that in many disciplines the number of objective function evaluations may be a preferred metric. Due to the approach, this may be approximated as $p \times g_p$.

2. Time taken per run of the algorithm on the indicated experiment. All times given are for running the EA omitting initialisation.
3. The RMS error (10) between the efficient frontier found by QP and that obtained by the EA after $g = 2.5nm$ generations (this value was obtained by experimentation and based on the first metric above). This differs from the $g = 1000n$ criterion of [7] (and [17] for the constrained problem) in the number of generations required and the recognition of runtime dependence on m.

As a basic measure of central tendency, we record the means of each of the three above metrics. However, it is acknowledged that the distribution of output metrics from an EA (being stochastic) is often characterised by a large dispersion of outcomes and so using mean (and, commonly, standard deviation) does not give a complete picture of algorithmic behaviour. Thus, to communicate the range of the distributions of RMSE and objective value across the runs, we give the minimum and maximum of the third metric as well as the final objective value, C_g, reached (which is, of course, related to the first metric). This allows us to present convergence analysis and also to be confident of the range of results we expect from the EA for a given dataset and value of m (see [25] for a fuller explanation).

Criteria such as a target RMSE (based upon a percentage of the maximum and minimum average returns associated to a given dataset) were attempted but found to be unreliable. Such criteria are felt to be overly simplistic, ignoring pertinent aspects of a given dataset. We found that large values of m are not necessary to produce the CCEF to an acceptable degree of precision. For example, $m = 10$ values of λ give a reasonable indication of the shape, with $m = 200$ values of λ giving particularly refined results. Larger values of m are possible, indicating the scope of the method but we omit results using such values of m in this work and settle for $m = 50$ for the purposes of speed. Due to the large number of experiments in this work, ten runs of each dataset were performed, with the above measures taken for each run.

4.2 EA Sensitivity to Initial Weights Matrix

It is generally acknowledged that EAs tend to be sensitive to initial values (however such values are defined). In our work, we ran tests to demonstrate the sensitivity of the EA to changes in initial weight matrix on the unconstrained problem (i.e., the value of the cardinality constraint was $k = n$). Three distinct types of initial weight matrix were created. The first type of matrix (named "Return Guess" in Table 3) was where every entry in each column of the initial weight matrix had value equal to the fraction of the total average return obtained from that asset. The second type was an educated guess of the following form. Let h refer to the number identifying an asset with largest average return. Then, after we let $\mathbf{W}_0 = \mathbf{0}_{m\times n}$, let

$$W_0(i,h) = 1 - \frac{i-1}{m-1} \text{ for } i = 1,\ldots,m, \tag{16}$$

be denoted by "linear", with the remaining weights allocated among another $k-1$ entries in that row at random. The third type stipulates that each row of the random weight matrix was independently generated as a random unit sum vector of length n (using the approach of [24]) conforming to the conditions of the applicable problem.

As above, ten runs of each experiment according to the initial weight matrices shown were performed and $m = 50$ values of λ were used.

Table 3: Results of varying the initial weights matrix. The notation g_p refers to the number of generations until 99.9% of the ultimate objective value is achieved (Section 4.1). A population size of 100 was used for accuracy.

Dataset	Return Guess		Linear		Random	
	Mean RMSE	g_p	Mean RMSE	g_p	Mean RMSE	g_p
D1	8.4314×10^{-5}	1969.1	7.6179×10^{-5}	1861.7	6.5015×10^{-5}	2136.1
D2	7.7405×10^{-5}	6362	6.1220×10^{-5}	5732	7.4603×10^{-5}	6657.7
D3	6.9021×10^{-5}	5844	3.6989×10^{-5}	5020.4	3.1122×10^{-5}	6214.8
D4	6.2850×10^{-5}	6664.6	5.1771×10^{-5}	5950.9	3.7896×10^{-5}	7008
D5	2.9861×10^{-5}	19260	2.9212×10^{-5}	16820.6	3.2444×10^{-5}	19249.4
Mean	6.4690×10^{-5}	8019.9	5.1074×10^{-5}	7083.1	4.8216×10^{-5}	8253.2

Observe from Table 3 that the EA using the linear type initial weights matrix consistently converges more quickly than the random and return guess types. The accuracy (mean RMSE) is also comparable with that obtained with the other two types of initial weights matrices. Hence, for all experiments we have chosen to use the linear type of initial weights matrix. The EA termination criterion was to exit the algorithm after the number of generations, g, given in the applicable cell and each experiment was performed with respect to the λ values specified in Table 2.

Next, we conduct an analysis of experimental data obtained by running the EA on the CCMPOP. Obtained results are given in the next section, after which a subseton of results are compared to the results of [4] in order to comment on EA accuracy.

5 Results

For the constrained problem, we used $m = 50$ values of λ and ran the EA for $g = 2.5nm = 125n$ generations to allow for result dependence on the value of k. This represents fewer iterations than the work of [7] and [17]. This section incorporates comparisons with results ran on the unconstrained MPOP; the only difference with runs here is that the EA was ran for $g = 2nm = 100n$ generations. This was sufficient for excellent results due to the small degree of problem difficulty and in this case the results are omitted. All datasets in this section were run on a Mac Pro workstation with a 3.7GHz 4-core Xeon E5 CPU and 16GB of RAM.

5.1 Size of Dominating Subset of Assets

The size, $k_{\min}$, of the dominating subset of assets for a given dataset is defined as the maximum number of assets in each row over all rows of the QP solution of the unconstrained MPOP which have non-trivial unconstrained QP weight (we take this to mean weights above a given tolerance, *tol*). Essentially, a dominating subset may

be seen as a subset of the assets which, somehow, tracks the behaviour of all the assets and is a set of the non-negligible QP weights for a given dataset. To compute these QP solutions, 2000 values of λ were used. Table 4 gives the dominating subset size for each dataset according to various tolerances. The values of $k_{\min}$ for (D1), (D4) and (D5) given in the table agree well with the maximal number of non-negligible weights exhibited in Figure 5 of [5].

Table 4: The sizes, $k_{\min}$, of dominating subset of each dataset, by (progressively smaller) tolerances. Observe that reducing the tolerance does not drastically affect $k_{\min}$ until between magnitudes -7 and -10. At these magnitudes, assets make barely perceptible contributions to objective value or RMSE measurements.

Tolerance *tol*	Dataset						
	D1	D2	D3	D4	D5	S&P389	S&P493
10^{-5}	12	26	34	39	14	39	37
10^{-6}	12	26	34	39	14	39	37
10^{-7}	12	26	34	40	14	39	38
10^{-10}	12	26	34	40	16	41	196
10^{-12}	16	85	55	53	73	274	493

We take $k = k_{\min}$, $tol = 10^{-5}$, $m = 50$ (as specified), $g = 2.5nm$ and run the EA on the CCMPOP. The summary statistics of these runs are given in Table 5 and are therein compared to those on the unconstrained problem. Note that identical results are not expected, since there may be weights of magnitude less than the tolerance with which the dominating subset of assets is chosen. All experiments in this table were performed with values of λ used as in Table 2.

Table 5: Results with $k = k_{\min}$. The notation ε_U denotes the mean RMSE between the EA unconstrained EF and the QPEF. The notation $\varepsilon_C(k_{\min})/\varepsilon_U$ denotes the ratio between $\varepsilon_C(k_{\min})$ in this table (the mean RMSE between the EA-computed CCEF with $k = k_{\min}$ and the QPEF) and the mean RMSE values from unconstrained runs.

Dataset	D1	D2	D3	D4	D5	S&P389	S&P493
ε_U	8.61e-5	7.83e-5	4.75e-5	7.59e-5	5.43e-5	3.92e-5	1.60e-5
Mean unc. obj. val.	-1.87e-3	-3.49e-3	-1.66e-3	-3.34e-3	-1.28e-3	-7.62e-4	-5.09e-4
$\varepsilon_C(k_{\min})$	3.09e-4	3.07e-4	1.76e-4	1.99e-4	3.59e-4	1.92e-4	7.00e-5
Mean const. obj. val.	-1.84e-3	-3.47e-3	-1.64e-3	-3.32e-3	-1.18e-3	-7.22e-4	-5.08e-4
g_p	2149.4	3129.5	4465.2	4304.2	6740.9	28972.4	14746
$\varepsilon_C(k_{\min})/\varepsilon_U$	3.59	3.92	3.71	2.62	6.61	4.90	4.38
Obj. val. ratio	0.984	0.994	0.988	0.994	0.921	0.948	0.997

Observe that the ratio $\varepsilon_C/\varepsilon_U$ is greater than 1, as would be expected. In addition, the ratio between mean constrained ($k = k_{\min}$) and unconstrained ($k = n$) objective values is strongly dependent upon the make-up of the portfolio. Note that, whilst the objective value ratio is generally close to 1 - dependent upon the dataset in question - the ratio of RMS errors is often much larger. The objective function is an average

of the objective values over a number of values of λ and is clearly less sensitive to changes in position of (risk, return) pairs than the RMS error. The next question we ask is whether the sizes of the dominating subsets of assets are replicated in EA experiments. Table 6 lists the size of the dominated subset of assets for typical EA solutions for a given dataset on the unconstrained MPOP, using fifty values of λ.

Table 6: The sizes, $u_{\min}$, of dominating subset of each dataset from the solutions of average EA runs on the unconstrained MPOP, i.e., the number of assets with weights at least *tol* in the final EA solutions.

Tolerance *tol*	Dataset						
	D1	D2	D3	D4	D5	S&P389	S&P493
10^{-5}	19	48	51	61	41	103	170
10^{-6}	24	61	60	74	78	153	237
10^{-7}	28	71	66	82	103	195	285
10^{-10}	31	82	83	97	156	300	399
10^{-12}	31	85	88	98	193	343	446

Clearly, the EA solutions have non-negligible contributions from a greater number of assets than do the QP solutions. In general, the results hint that 'good' approximations to the QPEF require $k > k_{\min}$. We now investigate this further, exploring the $\varepsilon_C/\varepsilon_U$ and objective value ratios for a range of k values either side of $k_{\min}$.

5.2 Testing Over a Range of Values of k

The discussion above shows that, in EA computations, finite contributions are often found from a far greater number of assets than might be expected by evaluating theoretical $k_{\min}$ values using QP. This raises a question of just how large the value of k should be to accurately predict the QPEF. In this section we run the EA on the CCMPOP for a range of k values for each dataset and compare with the unconstrained results. Our investigation results are given in Table 7 ($k \leq 60$) and Figure 4 ($k \leq 200$ for the largest datasets).

Table 7: Results of the EA on the CCMPOP for $m = 50$ values of λ and various k.

Dataset	k	Mean obj. val.	Min/Max obj. val.	Mean RMSE	Min/Max RMSE	g_p	Mean t_c
D1	5	-1.801e-3	-1.82e-3/-1.78e-3	5.79e-4	3.63e-4/7.53e-4	2688.1	260.5
	10	-1.832e-3	-1.85e-3/-1.82e-3	3.66e-4	2.39e-4/4.80e-4	2240.3	217.7
	15	-1.845e-3	-1.85e-3/-1.83e-3	2.96e-4	1.79e-4/3.92e-4	2028.2	201.9
D2	5	-3.446e-3	-3.47e-3/-3.42e-3	5.53e-4	3.34e-4/9.68e-4	4580.2	437.6
	10	-3.445e-3	-3.47e-3/-3.30e-3	4.38e-4	2.59e-4/8.00e-4	3474.6	333.3
	15	-3.462e-3	-3.48e-3/-3.44e-3	3.53e-4	2.56e-4/5.34e-4	3528.1	326.4
	20	-3.469e-3	-3.47e-3/-3.46e-3	3.22e-4	2.37e-4/4.27e-4	3158.4	317.8
	30	-3.473e-3	-3.48e-3/-3.47e-3	2.88e-4	1.80e-4/5.07e-4	3937.0	346.7
	45	-3.483e-3	-3.49e-3/-3.48e-3	2.07e-4	7.20e-5/3.28e-4	2901.1	255.9
	60	-3.488e-3	-3.49e-3/-3.49e-3	1.45e-4	6.26e-5/2.75e-4	3727.3	328.8
D3	5	-1.606e-3	-1.63e-3/-1.60e-3	3.87e-4	2.61e-4/5.5e-4	5523.6	594.7
	10	-1.621e-3	-1.63e-3/-1.61e-3	2.81e-4	1.85e-4/4.19e-4	5072.7	486.2
	15	-1.628e-3	-1.64e-3/-1.62e-3	2.21e-4	1.69e-4/2.84e-4	6738.4	621.7
	20	-1.633e-3	-1.64e-3/-1.63e-3	2.19e-4	1.69e-4/2.82e-4	5365.5	477.0
	30	-1.640e-3	-1.65e-3/-1.64e-3	1.67e-4	1.15e-4/2.08e-4	4305.9	381.6
	45	-1.648e-3	-1.65e-3/-1.65e-3	1.41e-4	1.10e-4/1.93e-4	3372.3	298.3
	60	-1.655e-3	-1.66e-3/-1.65e-3	8.47e-5	6.79e-5/1.23e-4	3829.5	338.8
D4	5	-3.207e-3	-3.28e-3/-3.08e-3	6.08e-4	2.51e-4/1.07e-3	5990.7	724.8
	10	-3.234e-3	-3.29e-3/-3.19e-3	5.00e-4	2.96e-4/6.91e-4	7135.8	695.9
	15	-3.269e-3	-3.31e-3/-3.22e-3	3.85e-4	2.71e-4/5.33e-4	8456.9	797.8
	20	-3.295e-3	-3.31e-3/-3.26e-3	3.15e-4	2.42e-4/4.55e-4	5648.0	506.8
	30	-3.306e-3	-3.32e-3/-3.30e-3	2.60e-4	1.91e-4/3.24e-4	5158.3	460.5
	45	-3.322e-3	-3.33e-3/-3.31e-3	1.78e-4	1.33e-4/2.20e-4	4167.6	371.5
	60	-3.332e-3	-3.34e-3/-3.33e-3	1.38e-4	9.22e-5/2.10e-4	3974.1	353.9
D5	5	-1.135e-3	-1.19e-3/-1.09e-3	4.87e-4	2.55e-4/6.41e-4	4239.1	721.9
	10	-1.181e-3	-1.22e-3/-1.12e-3	3.52e-4	2.02e-4/5.23e-4	7833.1	947.9
	15	-1.203e-3	-1.23e-3/-1.15e-3	3.34e-4	2.02e-4/4.38e-4	8888.0	1012.4
	20	-1.217e-3	-1.24e-3/-1.17e-3	2.79e-4	1.57e-4/4.02e-4	9767.8	1024.4
	30	-1.227e-3	-1.25e-3/-1.17e-3	2.32e-4	1.15e-4/3.32e-4	11277.9	1174.8
	45	-1.240e-3	-1.25e-3/-1.23e-3	2.21e-4	1.78e-4/2.66e-4	7490.5	775.0
	60	-1.249e-3	-1.26e-3/-1.24e-3	1.75e-4	1.08e-4/2.43e-4	8741.9	904.4
S&P389	5	-6.223e-4	-6.88e-4/-5.24e-4	7.59e-4	2.83e-4/1.41e-3	22424.8	3592.3
	10	-6.687e-4	-7.11e-4/-6.15e-4	4.52e-4	2.25e-4/8.70e-4	26374.2	3871.5
	15	-6.761e-4	-7.17e-4/-6.12e-4	4.96e-4	1.97e-4/9.33e-4	31960.8	4480.5
	20	-6.841e-4	-7.27e-4/-6.64e-4	3.82e-4	1.54e-4/7.25e-4	25579.9	3319.3
	30	-6.834e-4	-7.31e-4/-6.28e-4	4.16e-4	1.34e-4/7.69e-4	30349.0	3976.6
	45	-7.188e-4	-7.42e-4/-6.64e-4	2.28e-4	9.52e-5/7.37e-4	28333.7	3631.0
	60	-7.225e-4	-7.48e-4/-6.87e-4	2.39e-4	1.04e-4/7.84e-4	32714.4	4160.9
S&P493	5	-4.855e-4	-5.06e-4/-4.68e-4	2.03e-4	1.22e-4/3.10e-4	2059.6	516.7
	10	-4.964e-4	-5.08e-4/-4.75e-4	1.46e-4	5.30e-5/2.63e-4	3783.4	737.4
	15	-4.915e-4	-5.09e-4/-4.75e-4	1.53e-4	4.47e-5/2.53e-4	4941.6	920.6
	20	-5.038e-4	-5.08e-4/-4.77e-4	9.45e-5	5.24e-5/2.29e-4	8727.5	1308.5
	30	-5.026e-4	-5.09e-4/-4.81e-4	8.74e-5	4.35e-5/1.84e-4	3640.4	545.4
	45	-5.083e-4	-5.09e-4/-5.08e-4	6.84e-5	5.13e-5/9.33e-5	10780.8	1653.7
	60	-5.083e-4	-5.09e-4/-5.08e-4	4.63e-5	3.50e-5/6.93e-5	6318.1	918.0

It is clear that, for any given dataset, as $k \to n$ the mean objective value decreases uniformly (and further, the mean objective value approaches that of the unconstrained case). In this limit, the RMSE strictly decreases for the datasets (D1)–(D5) and generally decreases for the two S&P datasets.

5.3 Further Comparisons Between Constrained and Unconstrained Results

In this subsection we further compare constrained and unconstrained results, building upon the results of Table 7, with Figures 2–4 showing statistics for cardinality constraints up to $k = 200$. The data of Table 7, for values $5 \leq k \leq 60$ (for dataset (D1), $5 \leq k \leq 15$), form a subset of the data summarised in the figures. Figure 2 shows the objective value ratios for various values of k (that is, each value in the mean objective value column of Table 7 divided by the applicable mean objective value from the unconstrained problem on the vertical axis against n on the horizontal. This plot indicates the objective value ratio approaches one from below as k increases.

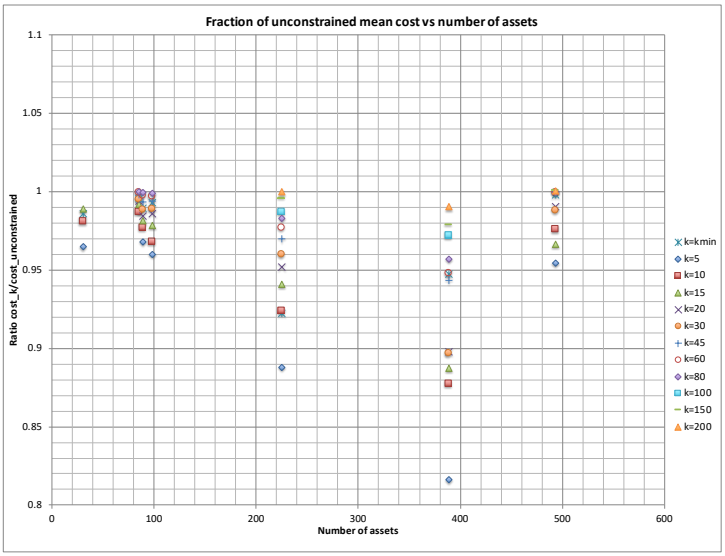


Fig. 2: Objective value ratios for various values of cardinality constraint, k.

Conversely, Figure 3 shows the analogous RMSE ratios for various values of k, illustrating that this ratio approaches one from above as k increases, albeit much more slowly than the previous figure. This provides further evidence that objective value and RMSE are fundamentally distinct metrics, from a quantitative as well as a qualitative perspective. From the above conclusion we infer that as k increases the RMSE decreases for a variety of datasets.

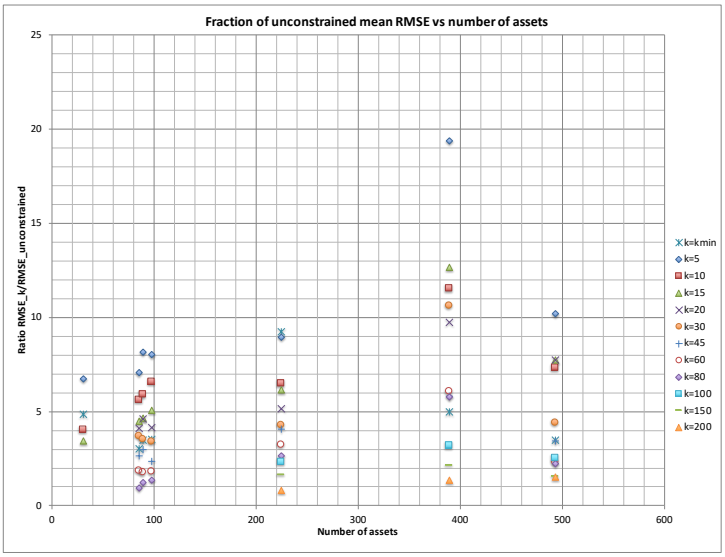


Fig. 3: RMSE ratio for various values of cardinality constraint, k.

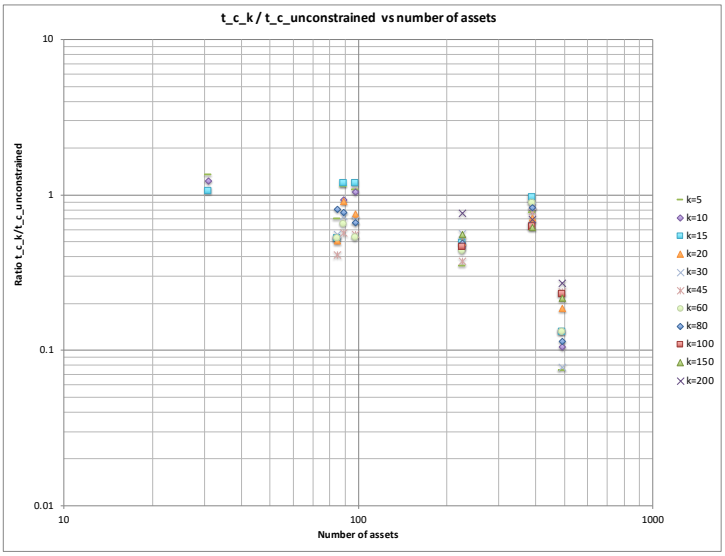
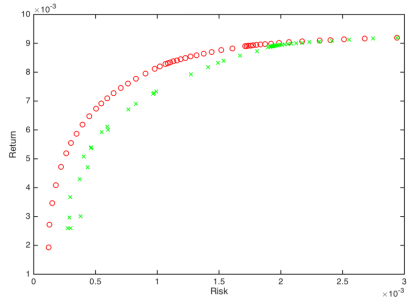


Fig. 4: Ratios of t_c for various values of cardinality constraint, k.

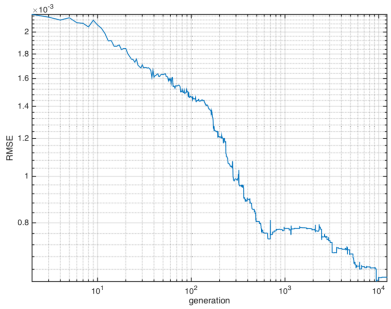
Finally, Figure 4 gives the analogous time to g_p-convergence ratios. This reveals an interesting property of the (D5) and S&P493 datasets: only a small fraction of the time to produce a solution to the unconstrained problem is needed to produce a solution of the constrained problem, even for large values of k (e.g., $k = 200$). This means not only that it may be possible to achieve results close, by our two metrics,

to the unconstrained problem by choosing an appropriate k, but also that this solution (and the respective convergence time to it) may be produced in a fraction of the time. However, this behaviour appears to be strongly dependent upon the portfolio composition because the convergence times for the S&P389 dataset for high values of k are much closer to the corresponding times for the unconstrained problem. Convergence times in general across all datasets are quite variable (Table 7, Figure 4).

Representative EA plots are given in Figure 5. Figure 5a shows a comparison of an obtained CCEF with the unconstrained QPEF. Figure 5b shows the evolution of RMSE between the QP and EA solution, and also the convergence to a minimal solution. The plot seems decidedly ragged and this is an indicator of problem difficulty. Indeed, the RMSE over subsequent generations in this figure is not strictly decreasing, meaning the EA seems to choose candidate EFs further away (in RMSE terms) from the (ideal) CCEF as they are preferable in objective value terms. (A cost-generation curve is omitted as the convergence is also illustrated by this subfigure.) In addition, Table 8 states the "ideal" minimal values of k using the criteria that the objective value ratio is greater than 0.99 and the RMSE ratio less than 2. Observe that these broadly agree with the values of $u_{\min}$ given in Table 6.



(a) Comparing QPEF (circles) with EA-obtained CCEF (crosses).



(b) RMSE between QPEF and CCEF for subsequent generations of a run.

Fig. 5: Representative plots for a run on the CCMPOP for the dataset (D4) and $k = 5$.

Table 8: Ideal k values which satisfy the given performance criteria for datasets (D2)–(D5), S&P389 and S&P493.

Dataset	D2	D3	D4	D5	S&P389	S&P493
n	85	89	98	225	389	493
Min k	45-60	45-60	45-60	100-150	∼200	100-150

To summarise, the EA produces very accurate solutions. The metrics of RMSE and objective value seem to be independent, but a trade-off between accuracy and time taken is achieved. However, the above comparisons between the EA-produced CCEF and the unconstrained QPEF are not the only possible comparisons. Further comments on accuracy are needed; in this spirit, we next give illustrative comparisons between our work and the results of [4].

5.4 Accuracy of Results

We use a sifting method of pooling our results to give a useful and more realistic comparison. The method is described below.

5.4.1 Methodology of Pooled Results

As discussed by [7], the 'λ-formulation' (Section 2.1) leads to gaps in the CCEF. This is due to the fact that the gradient of the efficient frontier corresponding to the λ-value λ_i is equal to $\lambda_i/(1-\lambda_i)$. For a sub-portfolio of k assets from a master portfolio of n assets, there are $n!/(n-k)!k!$ sub-EFs. The process of identifying (risk, return) points on the EF essentially samples from the sub-EFs and, for a given value of λ, the optimal (risk, return) point is the one that minimises the objective $\lambda_i V_i - (1-\lambda_i)R_i$. Given the vast number of possible sub-portfolios even if n and k are relatively modest (e.g. 44 million combinations for n = 31, k = 10) the algorithm may not in fact find the optimal EF point but might find a point on a sub-EF with a slightly higher objective value. Fortuitously, this point might occasionally lie close to a true EF point that is missed by the λ-formulation. This observation leads to a method for pooling results from several realisations to generate an approximate EF which can be a significant improvement over the EF generated from a single realisation. This process (Algorithm 5) is illustrated in Figure 6a, which shows EFs from individual realisations for $n = 89$ and $k = 5$. These results are then pooled and it is clear that some points are 'dominated' by others; i.e., these points have a lesser return and/or greater variance than non-dominated points. For each non-dominated point, there are no other points that have both greater return and lesser risk. The pooled points are then sifted (Figure 6b) to identify and remove all the dominated points and the remaining, non-dominated points form an improved EF approximation.

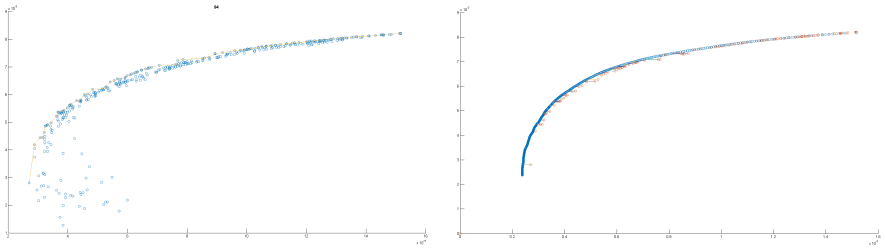
Algorithm 5 Pooling/sifting

input: Pooled n_{max} points from sub-EFs
output: Updated n_{max}: set of non-dominated EF points
$i \leftarrow 1$
while $i < n_{\max}$ **do**
  find $\max_j R(j)$, $j = i, \ldots, n_{\max}$
  swap $(V(\max_j), R(\max_j))$ with $(V(i), R(i))$
  for all $j = i+1, \ldots, n_{\max}$ **do**
    remove $(V(j), R(j))$ if $V(j) > V(i)$
    re-index remaining points $j > i+1$
    $n_{\max} \leftarrow n_{\max} - 1$
  end for
  $i \leftarrow i+1$
end while

To test the effectiveness of this strategy, in the following subsection, we compare the results obtained against the results found by [4] for several of the OR-Library datasets.

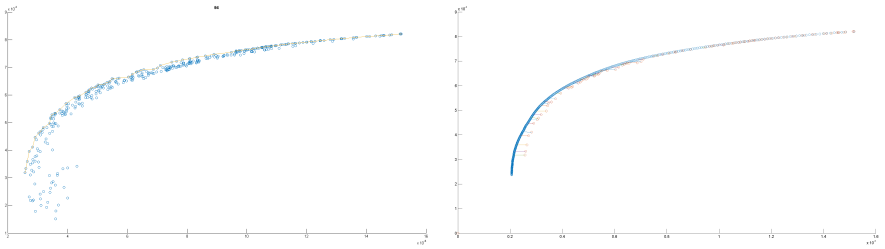
5.4.2 Summary of Results

The work of [4] characterises the CCPOP with minimum proportions as a two-objective formulation. That is, risk is minimised subject to a given level of return, each asset having a minimum proportion of 1%. This is performed through mapping the problem to a modified QP problem with added cardinality constraint and minimum proportion, and then solving using an adapted QP method. The formulation is a distinct one from our λ- formulation and has the advantage that no regions of the CCEF are inaccessible; i.e., there are no gaps in the generated CCEFs. A slight disadvantage is that the generated CCEF can be non-monotonic. To avoid this problem, the sifting process is applied to the CCEF of [4] to ensure that it is indeed monotonic. Figures 6b and 7b give a graphical comparison of our pooled results (dominated portfolios removed) with the CCEFs of [4] for dataset (D3) and cardinality constraints $k = 5$ and $k = 10$.



(a) "Before": Pooled (b) "After": Sifted

Fig. 6: Comparisons for dataset (D3) and cardinality $k = 5$. The left subfigure ("before") presents the pooled original results. The right subfigure ("after") compares the dominated pooled results and reference CCEF (blue circles denote the fixed reference CCEF, orange the pooled EA data with dominated portfolios removed).



(a) "Before": Pooled (b) "After": Sifted

Fig. 7: Comparisons for dataset (D3) and cardinality $k = 10$. Shown are the pooled original results (left), and the dominated pooled results and reference CCEFs (right).

It is clear by inspection that, by pooling and sifting the results, our method is able to approximate the above CCEFs very well. Figures 6a and 7a depict the original pooled EA results in each case. Note that the comparison is illustrative; based upon

experiments, it is believed the differences between CCEFs with or without minimum proportions of 1% is visually negligible for cardinalities $k = 5$ and $k = 10$.

Table 9 gives the RMS and mean absolute error between the pooled EA results and the results of [4]. The results indicate our work gives a good approximation to those of the authors, with the RMS errors an order of magnitude smaller than those reported in Table 7, i.e., prior to pooling/sifting. The mean absolute error broadly increases in line with the number of assets, n.

Table 9: Pooled data RMSEs and MAEs for (D1)–(D5), cardinalities $k = 5, 10$.

Dataset	$k = 5$		$k = 10$	
	RMSE	Mean abs. error	RMSE	Mean abs. error
D1	7.20e-5	5.09e-5	6.71e-5	4.07e-5
D2	7.04e-5	4.07e-5	7.12e-5	4.26e-5
D3	6.77e-5	4.17e-5	6.75e-5	4.14e-5
D4	7.68e-5	5.52e-5	7.51e-5	5.41e-5
D5	8.03e-5	6.21e-5	7.40e-5	5.28e-5
Mean	7.34e-5	5.01e-5	7.10e-5	4.63e-5

Note there are some other works (for example, [1]) which use a comparison with the CCEF of [5] (comparing several heuristic methods). However, in the cited work, no statistics on the error for (D1)–(D5) are present (although the authors do compare errors for two larger datasets). Furthermore, it is clear that the pooled methodology is valid for any risk where the return is an increasing function of risk. It is also possible to arrive at good results for even smaller cardinalities (the most difficult case of $k = 2$, for example). In the next section we conclude our work.

6 Conclusions and Further Work

This work gives a rigorous analytical and statistical treatment of a scalable method to approximate solutions to cardinality-constrained multiple POPs, distinguishing our treatment from "end result"-based approaches. Our method requires many fewer iterations to achieve good results in a reasonable time, representing greater efficiency than the detailed comparative approaches. Our approach also simultaneously, via the use of a matrix EA representation, computes varying numbers of portfolios. The objective function is averaged over a large number of λ values, but still discriminates efficiently between solutions at individual values of λ. From over 1500 runs of the EA it is illustrated that the EA results compare well with standard benchmark literature [7, 17] in terms of two independent metrics (RMSE and objective value), revealing promising results concerning accuracy and time savings.

The number, $k_{\min}$, of assets that are the most "important" in each benchmark data set was identified and experiments were run for values of k either side of this number. This shows that we may obtain increasingly close approximation of the QPEF

by increasing the value of k. Of course, this is not the whole story. The results and discussion of Section 5 show that a useful way of saving computational cost (and thus time) for large datasets is to run with $k < n$ but somehow "large enough" to produce a solution sufficiently close to the unconstrained one (both in terms of objective value and RMSE). Ranges of values of k are given for each relevant dataset, under which the objective value and RMSE achieve a close approximation of the generated CCEF with the QPEF. Finally, by pooling results and using a sifting algorithm to produce a non-dominated EF, we have also shown that our results compare well with the CCEFs established by [4].

We acknowledge the limitations of this work. For instance, the time complexity of matrix multiplication involved in computing the objective function for m simultaneous portfolios may be an issue, despite the benefits. Further work to remedy this may involve dividing the dataset into smaller sub-datasets consisting of contiguous sequences of rows of the weights matrix (partitioning the range of λ values into subsets), giving smaller matrices to process and allowing larger datasets to be tackled. Interim experiments indicate that taking such an approach could cut the computational cost by a factor of five for the largest datasets for the unconstrained problem. However, there is a balance to find the "sweet spot" between the extremes of computing portfolios individually and computing ever-larger matrices of multiple portfolios. Intuitively, this approach would also be fruitful for the constrained problem but, to our information, an estimate of such a factor in this case is unknown.

As further work, the method presented is inherently applicable to high performance computing via farming with large numbers of cores, potentially producing large improvements in convergence time. The code is also expandable to deal with other constraints such as minimum/maximum proportions and minimum lots. In addition, one of our goals is to compute (and integrate) measures of risk based on generalised probabilistic models of the stock market in order to compare the portfolios found with those produced above. Such measures would come into play when considering (especially post-2008 global financial crisis) real-world return distributions of financial assets.

Funding

The authors have received no funding for the study.

Conflict of Interest

The authors declare that they have no conflict of interest.

Acknowledgements The authors would like to thank the Centre for Mathematical Sciences at the University of Plymouth, UK, for research support during preparation of this work.

References

1. Anagnostopoulos K, Mamanis G (2011) The mean–variance cardinality constrained portfolio optimization problem: An experimental evaluation of five multiobjective evolutionary algorithms. Expert Systems with Applications 38(11):14208–14217
2. Beasley JE (2000) Or-library URL `http://people.brunel.ac.uk/~mastjjb/jeb/orlib/portinfo.html`
3. Best MJ, Grauer RR (1991) On the sensitivity of mean-variance-efficient portfolios to changes in asset means: some analytical and computational results. Review of Financial Studies 4(2):315–342
4. Cesarone F, Scozzari A, Tardella F (2010) Efficient algorithms for mean-variance [ortfolio optimization with hard real-world constraints. Giornale dell'Istituto Italiano degli Attuari 72:37–56
5. Cesarone F, Scozzari A, Tardella F (2010) Solutions to portfolio problems in beasley's or-library URL `http://host.uniroma3.it/docenti/cesarone/datasetsw3_tardella.html`
6. Cesarone F, Scozzari A, Tardella F (2011) Portfolio selection problems in practice: a comparison between linear and quadratic optimization models. arXiv preprint arXiv:11053594
7. Chang TJ, Meade N, Beasley JE, Sharaiha YM (2000) Heuristics for cardinality constrained portfolio optimisation. Computers & Operations Research 27(13):1271–1302
8. Cohen I (2006) "fast algorithm for generating doubly-stochastic matrices". URL `http://www.mathworks.com/matlabcentral/newsreader/view_thread/132415`, date accessed: 09/11/16
9. Craven MJ, Jimbo HC (2012) Evolutionary algorithm solution of the multiple conjugacy search problem in groups, and its applications to cryptography. Groups-Complexity-Cryptology 4(1):135–165
10. Craven MJ, Jimbo HC (2013) An ea for portfolio selection over multiple investment periods with exponential transaction costs. In: Proceedings of the 15th annual conference companion on Genetic and evolutionary computation, ACM, pp 115–116
11. Cvitanić J, Karatzas I (1996) Hedging and portfolio optimization under transaction costs: a martingale approach. Mathematical finance 6(2):133–165
12. Duran FEC, Cotta C, Fernández-Leiva AJ (2012) A comparative study of multiobjective evolutionary algorithms to optimize the selection of investment portfolios with cardinality constraints. In: European Conference on the Applications of Evolutionary Computation, Springer, pp 165–173
13. Goldberg D (1989) Genetic algorithms in search, optimization, and machine learning. Addison Wesley, Reading, Massachusetts
14. Hirschberger M, Qi Y, Steuer RE (2010) Large-scale mv efficient frontier computation via a procedure of parametric quadratic programming. European Journal of Operational Research 204(3):581–588
15. Kolm PN, Tütüncü R, Fabozzi FJ (2014) 60 years of portfolio optimization: Practical challenges and current trends. European Journal of Operational Research

234(2):356–371
16. Koza JR (2010) Human-competitive results produced by genetic programming. Genetic Programming and Evolvable Machines 11(3-4):251–284
17. Lwin K, Qu R, Kendall G (2014) A learning-guided multi-objective evolutionary algorithm for constrained portfolio optimization. Applied Soft Computing 24:757–772
18. Mansini R, Speranza MG (1999) Heuristic algorithms for the portfolio selection problem with minimum transaction lots. European Journal of Operational Research 114(2):219–233
19. Markowitz H (1952) Portfolio selection. The journal of finance 7(1):77–91
20. Pafka S, Kondor I (2003) Noisy covariance matrices and portfolio optimization ii. Physica A: Statistical Mechanics and its Applications 319:487–494
21. QuantQuote (2013) Quantquote free historical stock data. URL `https://quantquote.com/historical-stock-data`, accessed 21/04/16
22. Samuelson PA (1969) Lifetime portfolio selection by dynamic stochastic programming. The review of economics and statistics pp 239–246
23. Sornette D, Simonetti P, Andersen JV (2000) φ q-field theory for portfolio optimization:fat tails and nonlinear correlations. Physics Reports 335(2):19–92
24. Stafford R (2006) "randfixedsum". URL `http://uk.mathworks.com/matlabcentral/fileexchange/9700-random-vectors-with-fixed-sum/content/randfixedsum.m`
25. Thomaidis NS, Vassiliadis V (2013) Stochastic convergence analysis of metaheuristic optimisation techniques. In: Towards Advanced Data Analysis, STUDFUZZ 285, Springer, pp 343–357
26. Vassiliadis V, Thomaidis N, Dounias G (2011) On the performance and convergence properties of hybrid intelligent schemes: application on portfolio optimization domain. In: European Conference on the Applications of Evolutionary Computation, Springer, pp 131–140